TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 14, No. 3, June 2015, pp. 534 ~ 542
DOI: 10.11591/telkomnika.v14i3.7798

High Performance Computing Clusters Design and Analysis Using Red Hat Enterprise Linux

Atiqur Rahman
Department of Computer Science & Engineering, University of Chittagong, Chittagong, Bangladesh
E-mail: atiqcse09@cu.ac.bd

Abstract

The purpose of this paper was to configure a cluster computing system to improve performance over that of a single computer, while typically being much more cost-effective than single computers of comparable speed. High performance computing more and more issued in all kinds of fields. But, for the common user, there are several questions on implementing the process of competing on the specialized clusters: such as expense high, management difficulty and operation complex and so on. To overcome these problems, this article designs and realizes the high performance computing environment based on Linux cluster after studying the key technology of cluster and parallel computing. Finally, the performance of system environment is tested by interactive version of CPI algorithm. Before discussion on HPC the cluster computing system with its classification, the advantages and application of clustering system will also be discussed here.

Keywords: *HPCC-high performance computing cluster**,** NTP-network time protocol, SSH-secure shell**,** NFS-network file share**,** PDSH-public domain super hero*

1. Cluster Computing & Linux

A supercomputer is one of the biggest, fastest computers right this minute. So, the definition of supercomputing is constantly changing. Supercomputing is also called High Performance Computing (HPC). High-Performance Computing (HPC) is a branch of computer science that focuses on developing supercomputers, parallel processing algorithms, and related software [6]. It has brought service demand increases in line shape with network high speed advancement. Server can overload in very short time when visiting demand increasing. Therefore, clusters technology emerges as the times require from this. At present, clusters system has applied in many fields, such as scientific research calculation, petroleum exploration, weather forecast and biological information, signal handling and so on. What we can foresee, along with symmetrically multiprocessing machine product largely used and high performance network product perfected, as well as various software and hardware production produced, systematic and application software emerged, new generation high performance clusters system will become popular platform of computer field.

1.1. Why Linux?

Although clustering can be performed on various operating systems like Windows, Macintosh, Solaris etc. , Linux has its own advantages which are as follows:

a) Linux runs on a wide range of hardware.
b) Linux is exceptionally stable.
c) Linux source code is freely distributed.
d) Linux is relatively virus free.
e) Having a wide variety of tools and applications for free.
f) Good environment for developing cluster infrastructure.

In the next chapter we will discussed about the overview of HPC cluster along with its features, uses and benefits. The rest of the paper will be organized according to the following structure: section 2 illustrated HPC cluster’s framework along with its features, benefits and uses, section 3 different steps of implementation of HPC cluster, section 4 benchmarking, results and performance analysis of HPC cluster and section 5 concludes this paper.

Received February 6, 2015; Revised April 13, 2015; Accepted May 7, 2015

2. Overview of High Performance Computing Cluster [6]

2.1. What is High Performance Computing Cluster?

High-Performance Computing (HPC) is used to describe computing environments which utilize supercomputers and computer clusters to address complex computational requirements, support applications with significant processing time requirements, or require processing of significant amounts of data. High-performance computing cluster use cluster nodes to perform concurrent calculations. A high-performance cluster allows applications to work in parallel, therefore enhancing the performance of the application. High performance cluster are also referred to as computational cluster or grid cluster of HPC cluster.

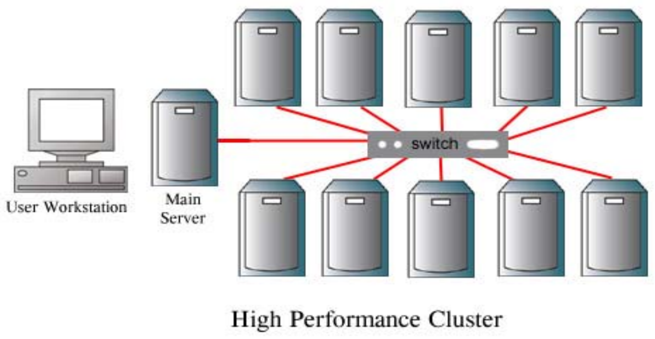


Figure1. Graphical View of High Performance cluster

2.2. Cluster Framework

Figure 2 shows the structure of a cluster.

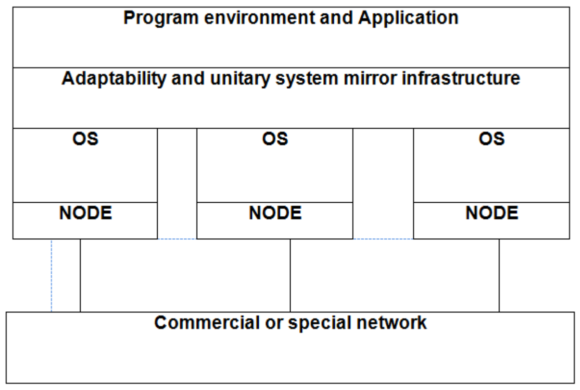

Program environment and Application
Adaptability and unitary system mirror infrastructure
OS
OS
OS
NODE
NODE
NODE
Commercial or special network

Figure 2. Cluster Framework

A cluster contains several of servers (at least two) which possess shared-data stock space. When any server runs an application, application data have been saved in the data space that shared. Operating system of server and file of application program of each server stores in local storage space. Each node in cluster server communicates with each other through internal area network. When a node of server occur fault, the running application program on this server will be taken over automatically by another node server.

Adopting the cluster architecture, there are many advantages such as free expansion, highly manageability, highly usability, high ratio of performance-to-price and so on, which solved the technology application task of crossing platform and operating system management, system software -hardware running state monitoring. It's ideal and strong system platform to undertake large-scale scientific project calculation.

2.3. Unique Benefits of HPC Clusters

The significance of high-performance computing (HPC) clustering is growing at a faster speed today. This is because more and more technical and scientific concerns are being analyzed on computer simulation. High performance computing is that using some processors parallel implements a task, which raises the efficiency of calculation. HPC clusters offer engineers, scientists, and technology analysts the required computing resources for making crucial decisions. This is required to promote product innovation, accelerate the pace of development, research and minimize the time to market. Leading service providers in this domain are supporting the R & D community by offering validated integrated solutions with optimized cluster configuration for certain applications.

3. Implementation of HPC Cluster [18]

3.1. Different Steps of HPC Cluster Implementation

Once Linux has been installed (preferably the same version) on all the nodes that need to be clustered. To realize the environment of parallel computing, we adopt four common PCs in design, select one of them as main supervision node computer that name is **hpcnd1** and IP address is **192.168.1.10**. The other three PCs are regarded as subordinate node computers that their name are **hpcnd2**, **hpcnd3** and **hpcnd4**, IP addresses are respectively **192.168.1.20, 192.168.1.30**and **192.168.1.40**. All of these are connected through network which adopts NETGEAR FS-608 version 2 switcher and formed star shape LAN structure. Install Linux operating system and the necessary tools package on each PC of four computer nodes (notice that close the firewall).

3.1.1. SSH (Secure Shell) Configuration

A packet-based binary protocol that provides encrypted connections to remote hosts or servers. Secure Shell is a program to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another. It provides strong authentication and secure communications over insecure channels. It is a replacement for rlogin, rsh, rcp, and rdist, telnet, ftp.

Connection to hpcnd2 closed.

The similar procedure will also have to apply respectively to the pair **hpcnd1-hpcnd3** and **hpcnd1-hpcnd4**, so that, the server (hpcnd1) can login into other clients securely without password button using a public key (using DSA-algorithm).

3.1.2. NTP (Network Time Protocol) Configuration

NTP stands for Network Time Protocol, and it is an Internet protocol used to synchronize the clocks of computers to some-time reference. NTP is an Internet standard protocol originally developed by Professor David L. Mills at the University of Delaware. Time usually just advances. If you have communicating programs running on different computers, time still should even advance if you switch from one computer to another. Obviously if one system is ahead of the others, the others are behind that particular one. From the perspective of an external observer, switching between these systems would cause time to jump forward and back, a non-desirable effect.

3.1.3. PDSH (Public Domain Super Heroes)

PDSH is an efficient, multithreaded remote shell client which executes commands on multiple remote hosts in parallel. Unlike rsh (remote shell) which runs commands on a single remote host, PDSH can run multiple remote commands in parallel. It is a threaded application that uses a sliding window (or *fan-out*) of threads to conserve resources on the initiating host and allow some connections to time out while all other connections continue.

3.1.4. NFS (Network File System) Configuration

NFS stands for Network File System, a file system developed by Sun Microsystems, Inc. It is a client/server system that allows users to access files across a network and treats them as if they resided in a local file directory. For example, if you were using a computer linked to a second computer via NFS, you could access files on the second computer as if they resided in a directory on the first computer. This is accomplished through the processes of exporting (the process by which an NFS server provides remote clients with access to its files) and mounting (the process by which file systems are made available to the operating system and the user). Simply we can say that NFS is a file system used for sharing of files over a network. Other resources like printers and storage devices can also be shared. This means that using NFS files can be accessed remotely.

3.1.5. MPICH2 Installation & Configuration

MPICH2 is a portable implementation of MPI (Message Passing Interface), a standard for message-passing for distributed memory applications used in parallel computing. It provides an MPI implementation that efficiently supports different computation and communication platform including commodity clusters, high-speed networks and propriety high-end computing systems. MPICH2 is Free Software and is available for most flavors of UNIX and Microsoft Windows. MPICH2 provides a separation of process management and communication. The default runtime environment consists of a set of daemons, called mpd (music player daemon), that establish communication among the machines to be used before application process startup, thus providing a clearer picture of what is wrong when communication cannot be established and providing a fast and scalable startup mechanism when parallel jobs are started.

4. Benchmarking, Results & Performance Analysis

4.1. Benchmarking [19]

Benchmarking is a systematic process for identifying and implementing best or better practices. Dimensions typically measured are quality, time and cost.

Now let's start some benchmarking:

```
[hpower@hpcc1 cluster]$ cd mpich2-1.0.8p1/examples/
cpi file already complied and executeable
-rw-r--r-- 1 hpower power 678 Nov 3 2007 child.c
-rwxr-xr-x 1 hpower power 576450 Jun 6 14:34 cpi
-rw-r--r-- 1 hpower power 1515 Nov 3 2007 cpi.c
-rw-r--r-- 1 hpower power 1964 Jun 6 14:34 cpi.o
-rw-r--r-- 1 hpower power 4469 Nov 3 2007 cpi.vcproj
drwxr-xr-x 2 hpower power 4096 Jun 6 14:31 cxx
drwxr-xr-x 2 hpower power 4096 Mar 27 01:40 developers
-rw-r--r-- 1 hpower power 10446 Nov 3 2007 examples.sln
drwxr-xr-x 2 hpower power 4096 Jun 6 14:31 f77
drwxr-xr-x 2 hpower power 4096 Jun 6 14:31 f90
-rw-r--r-- 1 hpower power 455 Nov 3 2007 hellow.c
-rw-r--r-- 1 hpower power 1892 Nov 3 2007 icpi.c
-rw-r--r-- 1 hpower power 6802 Jun 6 14:31 Makefile
-rw-r--r-- 1 hpower power 6767 Mar 27 01:40 Makefile.in
-rw-r--r-- 1 hpower power 1490 Mar 12 2008 Makefile.sm
drwxr-xr-x 2 hpower power 4096 Mar 27 01:39 mpiexec
-rw-r--r-- 1 hpower power 1049 Nov 3 2007 parent.c
-rw-r--r-- 1 hpower power 46399 Nov 3 2007 pmandel.c
-rw-r--r-- 1 hpower power 47798 Nov 3 2007 pmandel_fence.c
-rw-r--r-- 1 hpower power 4522 Nov 3 2007 pmandel_fence.vcproj
-rw-r--r-- 1 hpower power 45576 Nov 3 2007 pmandel_service.c
-rw-r--r-- 1 hpower power 4532 Nov 3 2007 pmandel_service.vcproj
-rw-r--r-- 1 hpower power 47510 Nov 3 2007 pmandel_spaserv.c
-rw-r--r-- 1 hpower power 4510 Nov 3 2007 pmandel_spaserv.vcproj
```

From these examples we'll use icpi.c to which is the iterative version of cpi (cycles per instruction).

4.2. Results

To find the results as well as to test the performance of HPC cluster at first we have to run or compile the icpi.c program at the cluster. To compile the program we have to write the following commands:

```
[hpower@hpcnd1 ~]$ whichmpiexec
/cluster/mpich2/bin/mpiexec
[hpower@hpcnd1 examples]$ mpicc -o icpiicpi.c
```

Now execute it in different nodes and also open top -c to check the processes.

First we will run with single node:

```
[hpower@hpcnd1 ~]$ /cluster/mpich2/bin/mpiexec -n 1 /cluster/mpich2-1.0.8p1/examples/icpi
Or
[hpower@hpcnd1 examples]$ mpiexec -n 1 ./icpi
Enter the number of intervals: (0 quits) 1000000000
pi is approximately 3.1415926535921401, Error is 0.0000000000023470
wall clock time = 19.196214
Enter the number of intervals: (0 quits)

top -c output :
[root@hpcnd1 ~]# top -c
top - 15:53:08 up 1:57, 3 users, load average: 0.21, 0.11, 0.03
Tasks: 188 total, 2 running, 186 sleeping, 0 stopped, 0 zombie
Cpu(s): 12.5%us, 0.0%sy, 0.0%ni, 87.4%id, 0.1%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 3368196k total, 936760k used, 2431436k free, 140504k buffers
Swap: 12289716k total, 0k used, 12289716k free, 634932k cached
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
14289 hpower 25 0 2332 812 676 R 100 0.0 0:07.80./icpi
1 root 15 0 2032 640 552 S 0 0.0 0:01.82 init [5]
```

Let’s run with two nodes:

```
[hpower@hpcnd1 examples]$ mpiexec -n 2 ./icpi
Enter the number of intervals: (0 quits) 1000000000
pi is approximately 3.1415926535905170, Error is 0.0000000000007239
wall clock time = 10.303831
Enter the number of intervals: (0 quits)
hpcnd1 top -c Output:
[root@hpcnd1 ~]# top -c
top - 15:56:46 up 2:01, 3 users, load average: 0.14, 0.10, 0.04
Tasks: 188 total, 2 running, 186 sleeping, 0 stopped, 0 zombie
Cpu(s): 12.5%us, 0.0%sy, 0.0%ni, 87.4%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 3368196k total, 941472k used, 2426724k free, 140656k buffers
Swap: 12289716k total, 0k used, 12289716k free, 634876k cached
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
14312 hpower 25 0 2328 848 712 R 100 0.0 0:08.84 ./icpi
1 root 15 0 2032 640 552 S 0 0.0 0:01.82 init [5]
```

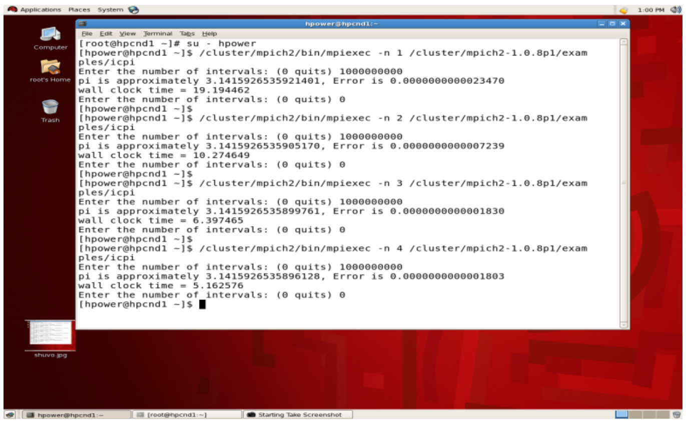
Let’s run with three nodes:

```
[hpower@hpcnd1 examples]$ mpiexec -n 3 ./icpi
Enter the number of intervals: (0 quits) 1000000000
pi is approximately 3.1415926535905170, Error is 0.0000000000007239
wall clock time = 6.869343
Enter the number of intervals: (0 quits)
```

Let's run with four nodes:

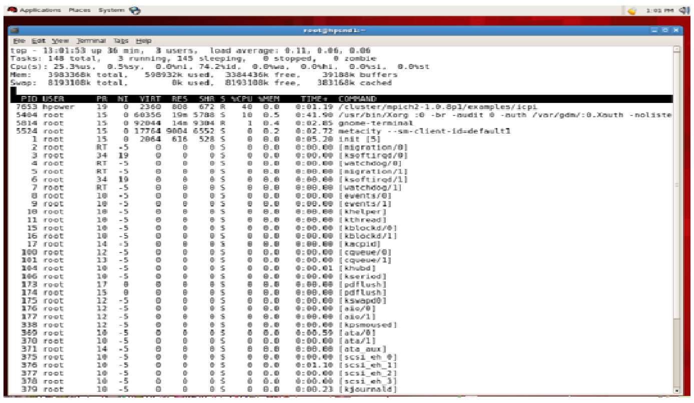
```
[hpower@hpcnd1 examples]$ mpiexec -n 4 ./icpi
Enter the number of intervals: (0 quits) 1000000000
pi is approximately 3.1415926535905170, Error is 0.0000000000007239
```
wall clock time = 4.211989
```
Enter the number of intervals: (0 quits)
```

The following screen shot showing the results of icpi.c compilation, i.e. wall clock time using single and multiple nodes.



```
[root@hpcnd1 ~]# su - hpower
[hpower@hpcnd1 ~]$ /cluster/mpich2/bin/mpiexec -n 1 /cluster/mpich2-1.0.8p1/exam
ples/icpi
Enter the number of intervals: (0 quits) 1000000000
pi is approximately 3.1415926535921401, Error is 0.0000000000023470
wall clock time = 19.194462
Enter the number of intervals: (0 quits) 0
[hpower@hpcnd1 ~]$
[hpower@hpcnd1 ~]$ /cluster/mpich2/bin/mpiexec -n 2 /cluster/mpich2-1.0.8p1/exam
ples/icpi
Enter the number of intervals: (0 quits) 1000000000
pi is approximately 3.1415926535905170, Error is 0.0000000000007239
wall clock time = 10.274649
Enter the number of intervals: (0 quits) 0
[hpower@hpcnd1 ~]$
[hpower@hpcnd1 ~]$ /cluster/mpich2/bin/mpiexec -n 3 /cluster/mpich2-1.0.8p1/exam
ples/icpi
Enter the number of intervals: (0 quits) 1000000000
pi is approximately 3.1415926535899761, Error is 0.0000000000001830
wall clock time = 6.397465
Enter the number of intervals: (0 quits) 0
[hpower@hpcnd1 ~]$
[hpower@hpcnd1 ~]$ /cluster/mpich2/bin/mpiexec -n 4 /cluster/mpich2-1.0.8p1/exam
ples/icpi
Enter the number of intervals: (0 quits) 1000000000
pi is approximately 3.1415926535896128, Error is 0.0000000000001803
wall clock time = 5.162576
Enter the number of intervals: (0 quits) 0
[hpower@hpcnd1 ~]$
```

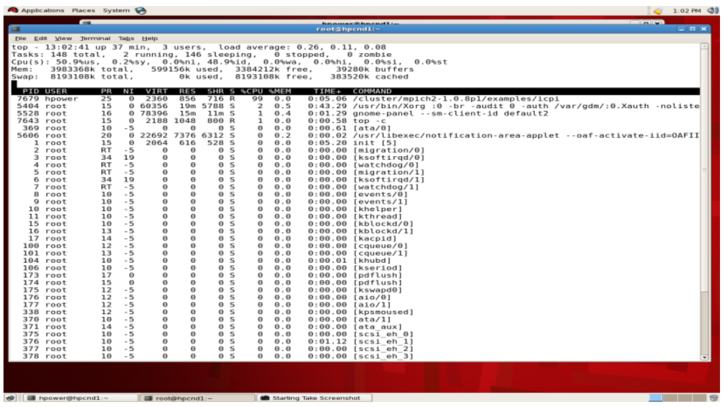
The following screen shoot showing the top –c output of each compilation.
1. Using single node (hpcnd1)



```
top - 13:01:53 up 36 min,  3 users,  load average: 0.11, 0.06, 0.06
Tasks: 148 total,   3 running, 145 sleeping,   0 stopped,   0 zombie
Cpu(s): 25.3%us,  0.5%sy,  0.0%ni, 74.2%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   3983368k total,   598932k used,  3384436k free,    39188k buffers
Swap:  8193108k total,        0k used,  8193108k free,   383168k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 7653 hpower    19   0  2360  808  672 R   40  0.0   0:01.19 /cluster/mpich2-1.0.8p1/examples/icpi
 5404 root      15   0 60356  19m 5788 S   10  0.5   0:41.90 /usr/bin/Xorg :0 -br -audit 0 -auth /var/gdm/:0.Xauth -noliste
 5814 root      15   0 92044  14m 9304 R    1  0.4   0:02.65 gnome-terminal
 5524 root      15   0 17764 9004 6552 S    0  0.2   0:02.72 metacity --sm-client-id=default1
    1 root      15   0  2064  616  528 S    0  0.0   0:05.20 init [5]
    2 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [migration/0]
    3 root      34  19     0    0    0 S    0  0.0   0:00.00 [ksoftirqd/0]
    4 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [watchdog/0]
    5 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [migration/1]
    6 root      34  19     0    0    0 S    0  0.0   0:00.00 [ksoftirqd/1]
    7 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [watchdog/1]
    8 root      10  -5     0    0    0 S    0  0.0   0:00.00 [events/0]
    9 root      10  -5     0    0    0 S    0  0.0   0:00.00 [events/1]
   10 root      10  -5     0    0    0 S    0  0.0   0:00.00 [khelper]
   11 root      10  -5     0    0    0 S    0  0.0   0:00.00 [kthread]
   15 root      10  -5     0    0    0 S    0  0.0   0:00.00 [kblockd/0]
   16 root      10  -5     0    0    0 S    0  0.0   0:00.00 [kblockd/1]
   17 root      14  -5     0    0    0 S    0  0.0   0:00.00 [kacpid]
  100 root      12  -5     0    0    0 S    0  0.0   0:00.00 [cqueue/0]
  101 root      13  -5     0    0    0 S    0  0.0   0:00.00 [cqueue/1]
  104 root      10  -5     0    0    0 S    0  0.0   0:00.01 [khubd]
  106 root      10  -5     0    0    0 S    0  0.0   0:00.00 [kseriod]
  173 root      17   0     0    0    0 S    0  0.0   0:00.00 [pdflush]
  174 root      15   0     0    0    0 S    0  0.0   0:00.00 [pdflush]
  175 root      12  -5     0    0    0 S    0  0.0   0:00.00 [kswapd0]
  176 root      12  -5     0    0    0 S    0  0.0   0:00.00 [aio/0]
  177 root      12  -5     0    0    0 S    0  0.0   0:00.00 [aio/1]
  338 root      12  -5     0    0    0 S    0  0.0   0:00.00 [kpsmoused]
  369 root      10  -5     0    0    0 S    0  0.0   0:00.59 [ata/0]
  370 root      10  -5     0    0    0 S    0  0.0   0:00.00 [ata/1]
  371 root      14  -5     0    0    0 S    0  0.0   0:00.00 [ata_aux]
  375 root      10  -5     0    0    0 S    0  0.0   0:00.00 [scsi_eh_0]
  376 root      10  -5     0    0    0 S    0  0.0   0:01.10 [scsi_eh_1]
  377 root      10  -5     0    0    0 S    0  0.0   0:00.00 [scsi_eh_2]
  378 root      10  -5     0    0    0 S    0  0.0   0:00.00 [scsi_eh_3]
  379 root      10  -5     0    0    0 S    0  0.0   0:00.23 [kjournald]
```

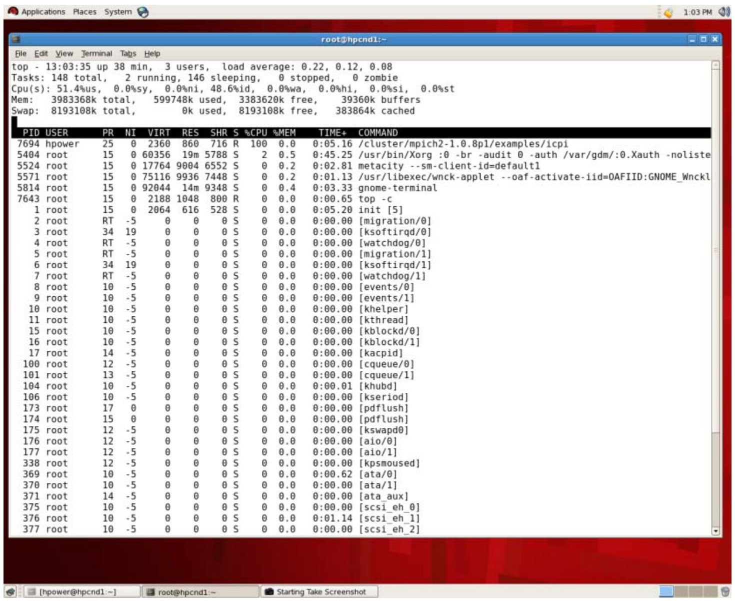
2. Using double nodes (hpcnd1+hpcnd3)



```
top - 13:02:41 up 37 min,  3 users,  load average: 0.26, 0.11, 0.08
Tasks: 148 total,   2 running, 146 sleeping,   0 stopped,   0 zombie
Cpu(s): 50.9%us,  0.2%sy,  0.0%ni, 48.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   3983368k total,   599156k used,  3384212k free,    39280k buffers
Swap:  8193108k total,        0k used,  8193108k free,   383520k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 7679 hpower    25   0  2360  856  716 R   99  0.0   0:05.06 /cluster/mpich2-1.0.8pl/examples/icpi
 5404 root      15   0 60356  19m 5788 S    2  0.5   0:43.29 /usr/bin/Xorg :0 -br -audit 0 -auth /var/gdm/:0.Xauth -noliste
 5528 root      16   0 78396  15m  11m S    1  0.4   0:01.29 gnome-panel --sm-client-id default2
 7643 root      15   0  2188 1048  800 R    1  0.0   0:00.58 top -c
  369 root      10  -5     0    0    0 S    0  0.0   0:00.61 [ata/0]
 5606 root      20   0 22692 7376 6312 S    0  0.2   0:00.02 /usr/libexec/notification-area-applet --oaf-activate-iid=OAFII
    1 root      15   0  2064  616  528 S    0  0.0   0:05.20 init [5]
    2 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [migration/0]
    3 root      34  19     0    0    0 S    0  0.0   0:00.00 [ksoftirqd/0]
    4 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [watchdog/0]
    5 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [migration/1]
    6 root      34  19     0    0    0 S    0  0.0   0:00.00 [ksoftirqd/1]
    7 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [watchdog/1]
    8 root      10  -5     0    0    0 S    0  0.0   0:00.00 [events/0]
    9 root      10  -5     0    0    0 S    0  0.0   0:00.00 [events/1]
   10 root      10  -5     0    0    0 S    0  0.0   0:00.00 [khelper]
   11 root      10  -5     0    0    0 S    0  0.0   0:00.00 [kthread]
   15 root      10  -5     0    0    0 S    0  0.0   0:00.00 [kblockd/0]
   16 root      13  -5     0    0    0 S    0  0.0   0:00.00 [kblockd/1]
   17 root      14  -5     0    0    0 S    0  0.0   0:00.00 [kacpid]
  100 root      12  -5     0    0    0 S    0  0.0   0:00.00 [cqueue/0]
  101 root      13  -5     0    0    0 S    0  0.0   0:00.00 [cqueue/1]
  104 root      10  -5     0    0    0 S    0  0.0   0:00.01 [khubd]
  106 root      10  -5     0    0    0 S    0  0.0   0:00.00 [kseriod]
  173 root      17   0     0    0    0 S    0  0.0   0:00.00 [pdflush]
  174 root      15   0     0    0    0 S    0  0.0   0:00.00 [pdflush]
  175 root      12  -5     0    0    0 S    0  0.0   0:00.00 [kswapd0]
  176 root      12  -5     0    0    0 S    0  0.0   0:00.00 [aio/0]
  177 root      12  -5     0    0    0 S    0  0.0   0:00.00 [aio/1]
  338 root      12  -5     0    0    0 S    0  0.0   0:00.00 [kpsmoused]
  370 root      10  -5     0    0    0 S    0  0.0   0:00.00 [ata/1]
  371 root      14  -5     0    0    0 S    0  0.0   0:00.00 [ata_aux]
  375 root      10  -5     0    0    0 S    0  0.0   0:00.00 [scsi_eh_0]
  376 root      10  -5     0    0    0 S    0  0.0   0:01.12 [scsi_eh_1]
  377 root      10  -5     0    0    0 S    0  0.0   0:00.00 [scsi_eh_2]
  378 root      10  -5     0    0    0 S    0  0.0   0:00.00 [scsi_eh_3]
```

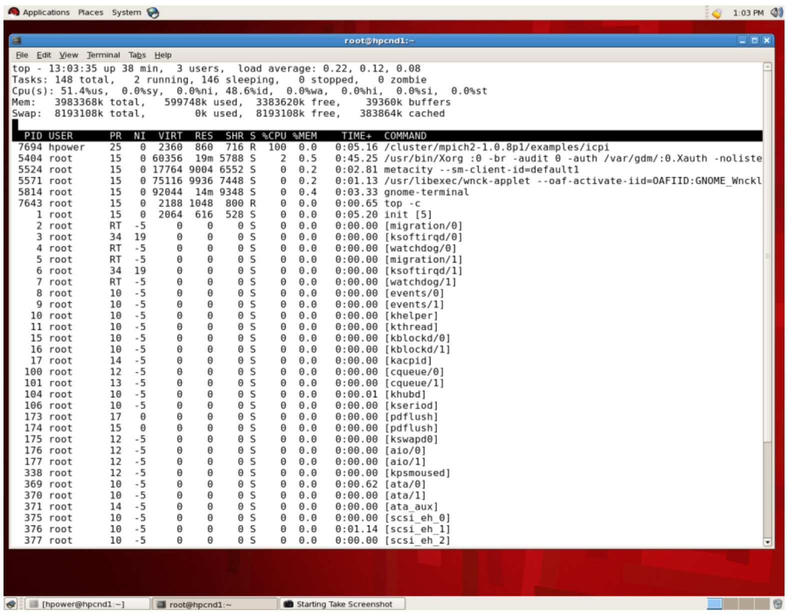
3. Using three nodes (hpcnd1+hpcnd2+hpcnd3)



```
top - 13:03:35 up 38 min,  3 users,  load average: 0.22, 0.12, 0.08
Tasks: 148 total,   2 running, 146 sleeping,   0 stopped,   0 zombie
Cpu(s): 51.4%us,  0.0%sy,  0.0%ni, 48.6%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   3983368k total,   599748k used,  3383620k free,    39360k buffers
Swap:  8193108k total,        0k used,  8193108k free,   383864k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 7694 hpower    25   0  2360  860  716 R  100  0.0   0:05.16 /cluster/mpich2-1.0.8p1/examples/icpi
 5404 root      15   0 60356  19m 5788 S    2  0.5   0:45.25 /usr/bin/Xorg :0 -br -audit 0 -auth /var/gdm/:0.Xauth -noliste
 5524 root      15   0 17764 9004 6552 S    0  0.2   0:02.81 metacity --sm-client-id=default1
 5571 root      15   0 75116 9936 7448 S    0  0.2   0:01.13 /usr/libexec/wnck-applet --oaf-activate-iid=OAFIID:GNOME_Wnckl
 5814 root      15   0 92044  14m 9348 S    0  0.4   0:03.33 gnome-terminal
 7643 root      15   0  2188 1048  800 R    0  0.0   0:00.65 top -c
    1 root      15   0  2064  616  528 S    0  0.0   0:05.20 init [5]
    2 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [migration/0]
    3 root      34  19     0    0    0 S    0  0.0   0:00.00 [ksoftirqd/0]
    4 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [watchdog/0]
    5 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [migration/1]
    6 root      34  19     0    0    0 S    0  0.0   0:00.00 [ksoftirqd/1]
    7 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [watchdog/1]
    8 root      10  -5     0    0    0 S    0  0.0   0:00.00 [events/0]
    9 root      10  -5     0    0    0 S    0  0.0   0:00.00 [events/1]
   10 root      10  -5     0    0    0 S    0  0.0   0:00.00 [khelper]
   11 root      10  -5     0    0    0 S    0  0.0   0:00.00 [kthread]
   15 root      10  -5     0    0    0 S    0  0.0   0:00.00 [kblockd/0]
   16 root      10  -5     0    0    0 S    0  0.0   0:00.00 [kblockd/1]
   17 root      14  -5     0    0    0 S    0  0.0   0:00.00 [kacpid]
  100 root      12  -5     0    0    0 S    0  0.0   0:00.00 [cqueue/0]
  101 root      13  -5     0    0    0 S    0  0.0   0:00.00 [cqueue/1]
  104 root      10  -5     0    0    0 S    0  0.0   0:00.01 [khubd]
  106 root      10  -5     0    0    0 S    0  0.0   0:00.00 [kseriod]
  173 root      17   0     0    0    0 S    0  0.0   0:00.00 [pdflush]
  174 root      15   0     0    0    0 S    0  0.0   0:00.00 [pdflush]
  175 root      12  -5     0    0    0 S    0  0.0   0:00.00 [kswapd0]
  176 root      12  -5     0    0    0 S    0  0.0   0:00.00 [aio/0]
  177 root      12  -5     0    0    0 S    0  0.0   0:00.00 [aio/1]
  338 root      12  -5     0    0    0 S    0  0.0   0:00.00 [kpsmoused]
  369 root      10  -5     0    0    0 S    0  0.0   0:00.62 [ata/0]
  370 root      10  -5     0    0    0 S    0  0.0   0:00.00 [ata/1]
  371 root      14  -5     0    0    0 S    0  0.0   0:00.00 [ata_aux]
  375 root      10  -5     0    0    0 S    0  0.0   0:00.00 [scsi_eh_0]
  376 root      10  -5     0    0    0 S    0  0.0   0:01.14 [scsi_eh_1]
  377 root      10  -5     0    0    0 S    0  0.0   0:00.00 [scsi_eh_2]
```

4. Using four nodes (hpcnd1+hpcnd2+hpcnd3+ hpcnd4)



```
top - 13:03:35 up 38 min,  3 users,  load average: 0.22, 0.12, 0.08
Tasks: 148 total,   2 running, 146 sleeping,   0 stopped,   0 zombie
Cpu(s): 51.4%us,  0.0%sy,  0.0%ni, 48.6%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   3983368k total,   599748k used,  3383620k free,    39360k buffers
Swap:  8193108k total,        0k used,  8193108k free,   383864k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 7694 hpower    25   0  2360  860  716 R  100  0.0   0:05.16 /cluster/mpich2-1.0.8p1/examples/icpi
 5404 root      15   0 60356  19m 5788 S    2  0.5   0:45.25 /usr/bin/Xorg :0 -br -audit 0 -auth /var/gdm/:0.Xauth -noliste
 5524 root      15   0 17764 9004 6552 S    0  0.2   0:02.81 metacity --sm-client-id=default1
 5571 root      15   0 75116 9936 7448 S    0  0.2   0:01.13 /usr/libexec/wnck-applet --oaf-activate-iid=OAFIID:GNOME_Wnckl
 5814 root      15   0 92044  14m 9348 S    0  0.4   0:03.33 gnome-terminal
 7643 root      15   0  2188 1048  800 R    0  0.0   0:00.65 top -c
    1 root      15   0  2064  616  528 S    0  0.0   0:05.20 init [5]
    2 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [migration/0]
    3 root      34  19     0    0    0 S    0  0.0   0:00.00 [ksoftirqd/0]
    4 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [watchdog/0]
    5 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [migration/1]
    6 root      34  19     0    0    0 S    0  0.0   0:00.00 [ksoftirqd/1]
    7 root      RT  -5     0    0    0 S    0  0.0   0:00.00 [watchdog/1]
    8 root      10  -5     0    0    0 S    0  0.0   0:00.00 [events/0]
    9 root      10  -5     0    0    0 S    0  0.0   0:00.00 [events/1]
   10 root      10  -5     0    0    0 S    0  0.0   0:00.00 [khelper]
   11 root      10  -5     0    0    0 S    0  0.0   0:00.00 [kthread]
   15 root      10  -5     0    0    0 S    0  0.0   0:00.00 [kblockd/0]
   16 root      10  -5     0    0    0 S    0  0.0   0:00.00 [kblockd/1]
   17 root      14  -5     0    0    0 S    0  0.0   0:00.00 [kacpid]
  100 root      12  -5     0    0    0 S    0  0.0   0:00.00 [cqueue/0]
  101 root      13  -5     0    0    0 S    0  0.0   0:00.00 [cqueue/1]
  104 root      10  -5     0    0    0 S    0  0.0   0:00.01 [khubd]
  106 root      10  -5     0    0    0 S    0  0.0   0:00.00 [kseriod]
  173 root      17   0     0    0    0 S    0  0.0   0:00.00 [pdflush]
  174 root      15   0     0    0    0 S    0  0.0   0:00.00 [pdflush]
  175 root      12  -5     0    0    0 S    0  0.0   0:00.00 [kswapd0]
  176 root      12  -5     0    0    0 S    0  0.0   0:00.00 [aio/0]
  177 root      12  -5     0    0    0 S    0  0.0   0:00.00 [aio/1]
  338 root      12  -5     0    0    0 S    0  0.0   0:00.00 [kpsmoused]
  369 root      10  -5     0    0    0 S    0  0.0   0:00.62 [ata/0]
  370 root      10  -5     0    0    0 S    0  0.0   0:00.00 [ata/1]
  371 root      14  -5     0    0    0 S    0  0.0   0:00.00 [ata_aux]
  375 root      10  -5     0    0    0 S    0  0.0   0:00.00 [scsi_eh_0]
  376 root      10  -5     0    0    0 S    0  0.0   0:01.14 [scsi_eh_1]
  377 root      10  -5     0    0    0 S    0  0.0   0:00.00 [scsi_eh_2]
```

4.3. Performance Analysis [20]

The following table showing the wall clock time after compiling the icpi.c using single or multiple nodes by using different intervals.

Table 1

No. of intervals(Cycles) Value of n in algorithm	100	10000	1000000	100000000	1000000000
hpcnd1 (wall clock time)	0.000036	0.000228	0.019338	1.920956	19.196214
hpcnd1+hpcnd2 (wall clock time)	0.002732	0.000927	0.010423	0.962485	10.303831
hpcnd1+hpcnd2+hpcnd3 (wall clock time)	0.005705	0.000766	0.007697	0.685562	6.869343
hpcnd1+hpcnd2+hpcnd3+hpcnd4 (wall clock time)	0.006224	0.000425	0.000259	0.425128	4.211989

Now we can represent the information of table with a graph to understand the performance analysis more easily through graphical representation.

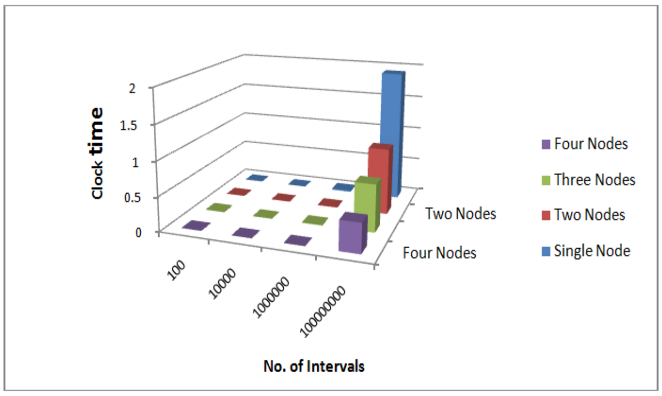


Figure 3. Graphical representation of number of intervals vs. wall clock time

5. Conclusion

With the popularity of high performance calculation and network technology, cluster system, which is the hot point of research and main stream of parallel computing in the world, is large advantage that can't be substituted. This paper offers the method of setting up the environment of parallel computing technology based on Linux and meets the needs of a large-scale scientific and engineering computing under the experimental environment. Here we have discussed the performance analysis of HPC cluster using only one algorithm, i.e. iterative cpi algorithm. There many others algorithm which are available for performance analysis. In future, the performance can be analyzed using other algorithms. The number nodes can also be increased to 8 to 16 nodes or more to improve the strength of HPC cluster. The HPC clusters can also be implemented using other version of Linux like Centos, Fedora or other operating system like UNIX or Mac etc.

References

[1] Yang, Shin-Jer, Chung-Chih Tu, Jyhjong Lin. *Design Issue and Performance Analysis of Data Migration Tool in a Cloud-Based Environment*. Proceedings of the 4th International Conference on Computer Engineering and Networks. Springer International Publishing. 2015.
[2] Hadjidoukas PE, et al. Π4U: A high performance computing framework for Bayesian uncertainty quantification of complex models. *Journal of Computational Physics.* 2015;284: 1-21.
[3] Yao, Yushu, et al. SciDB for High Performance Array-structured Science Data at NERSC. *Computing in Science & Engineering.* 2015; 1: 1-1.
[4] Younge, Andrew J, John Paul Walters, Geoffrey C Fox. Supporting High Performance Molecular Dynamics in Virtualized Clusters using IOMMU, SR-IOV, and GPUDirect. 2015.
[5] Visser, Marco D, et al. S1 Text: Speeding up ecological and evolutionary computations in R; essentials of high performance computing for biologists. 2015.
[6] Kaur, Arvinder, Shraddha Verma. Performance Measurement and Analysis of High-Availability Clusters. *ACM SIGSOFT Software Engineering Notes.* 2015; 40(2): 1-7.
[7] Goudey, Benjamin, et al. High performance computing enabling exhaustive analysis of higher order single nucleotide polymorphism interaction in Genome Wide Association Studies. *Health Information Science and Systems.* 2015; 1: 3.
[8] Yao, Yushu, et al. SciDB for High Performance Array-structured Science Data at NERSC. *Computing in Science & Engineering.* 2015; 1: 1-1.
[9] Belgacem, Mohamed Ben, Bastien Chopard. A hybrid HPC/cloud distributed infrastructure: Coupling EC2 cloud resources with HPC clusters to run large tightly coupled multiscale applications. *Future Generation Computer Systems.* 2015; 42: 11-21.
[10] Dumitrel Loghin, Bogdan Marius Tudor, et al. *A Performance Study of Big Data on Small Nodes*. Proceedings of the VLDB Endowment. 2015; 8(7).

[11] Hartog, Jessica, et al. Performance Analysis of Adapting a MapReduce Framework to Dynamically Accommodate Heterogeneity. *Transactions on Large-Scale Data-and Knowledge-Centered Systems XX*. Springer Berlin Heidelberg. 2015: 108-130.
[12] Du Jun, et al. Research on Linux-Based PC Cluster System and its Application in Numerical Simulation for Shallow Buried Soft Soil Tunnel. *Applied Mechanics and Materials*. 2015; 730
[13] Kobayashi Hiroaki. Feasibility study of a future HPC system for memory-intensive applications: final report. *Sustained Simulation Performance 2014*. Springer International Publishing. 2015: 3-16.
[14] Sharifi Hadi, Omar Aaziz, Jonathan Cook. *Monitoring HPC applications in the production environment*. Proceedings of the 2nd Workshop on Parallel Programming for Analytics Applications. ACM. 2015.
[15] Gudivada, Venkat N, Jagadeesh Nandigam, Jordan Paris. Programming Paradigms in High Performance Computing. *Research and Applications in Global Supercomputing.* 2015: 303.
[16] El-Moursy, Ali A, et al. Parallel PPI Prediction Performance Study on HPC Platforms. *Journal of Circuits, Systems and Computers.* 2015.
[17] Ahmed Munib, Ishfaq Ahmad, Mohammad Saad Ahmad. A survey of genome sequence assembly techniques and algorithms using high-performance computing. *The Journal of Supercomputing.* 2015; 71(1): 293-339.
[18] Islam Nusrat S, et al. *High performance RDMA-based design of HDFS over InfiniBand*. Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. IEEE Computer Society Press. 2012.
[19] Danalis Anthony, et al. *The scalable heterogeneous computing (SHOC) benchmark suite*. Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units. ACM. 2010.
[20] Nichols J, et al. HPC-EPIC for high resolution simulations of environmental and sustainability assessment. *Computers and Electronics in Agriculture.* 2011; 79(2): 112-115.
[21] Kopper Karl. The Linux Enterprise Cluster: build a highly available cluster with commodity hardware and free software. No Starch Press. 2005.
[22] Bookman Charles. Linux clustering: building and maintaining Linux clusters. Sams Publishing. 2003.
[23] Baker Mark. Cluster computing white paper. 2000.
[24] Yang, Chao-Tung, Yu-Lun Luo, Chuan-Lin Lai. Designing computing platform for BioGrid. *International journal of computer applications in technology.* 2005; 22(1): 3-13.